IN THE CLAIMS

Amended claims follow:

1. (currently amended) A computer program product for controlling a computer to detect an executable computer program containing a computer virus, said computer program product comprising:

analysis logic for analyzing program instructions forming said executable computer program to identify suspect program instructions being at least one of:

(i) a program instruction generating a result value not used by another portion of said executable computer program; and

(ii) a program instruction dependent upon an uninitialised variable; and

detecting logic for detecting said executable computer program as containing a computer virus if a number of suspect program instructions identified for said executable computer program exceeds a threshold level, wherein said analysis logic includes a dependence table indicating dependency between state variables within said computer and loaded variable values, and for each program instruction said analysis logic makes a determination as to which state variables are read and written by that program instruction and for each loaded variable value within said dependence table if any state variable read by that program instruction is marked as dependent upon said loaded variable value, then all state variables written by that program instruction are marked as dependent upon said loaded variable value with previous dependencies being cleared, and said analysis logic parses said executable computer program for suspect program instructions by following execution flow and upon occurrence of a branch first following a first branch path having

saved pending analysis results and subsequently returning to follow a second branch path having restored said pending analysis results;

wherein a state variable is marked as initialised upon occurrence of one of:

(i) a write to said state variable of a determined initialised value; and

(ii) use of said state variable as a memory address value by a program instruction;

wherein a branch path stops being followed when one of the following occurs:

(i) there are no further suitable program instruction for execution within that branch path; and

(ii) said branch path rejoins a previously parsed execution path;

wherein said analysis logic includes an initialisation table indicating which state variables have been initialised;

wherein said dependence table includes a column for each register within a processor, an external write, and a write to a flag value; and a plurality of rows with each row corresponding to a value loaded into said computer that influences a state of said computer;

wherein said initialisation table includes a column for each register within said processor indicating whether each register is initialised.

2. (original) A computer program product as claimed in claim 1, wherein said computer virus is a polymorphic computer virus.

3. (cancelled)

4. (currently amended) A computer program product as claimed in claim 1, wherein for each program instruction said analysis logic makes a determination as to which state variables are read by that program instruction to produce a read-mask.

5. (currently amended) A computer program product as claimed in claim 1, wherein for each program instruction said analysis logic makes a determination as to which state variables are written by that program instruction to produce a write-mask.

6. (cancelled)

7. (previously presented) A computer program product as claimed in claim 1, wherein said state variables include at least one of:

(i) register values;

(ii) processing result flag values; and

(iii) a flag indicative of a write to a non-register storage location.

8. (cancelled)

9. (cancelled)

10. (cancelled)

11. (cancelled)

12. (previously presented) A computer program product as claimed in claim 1, wherein if said threshold level is exceeded, then further virus detection mechanisms are triggered to confirm the presence of a computer virus.

13. (currently amended) A method of detecting an executable computer program containing a computer virus, said method comprising the steps of:

analysing program instructions forming said executable computer program to identify suspect program instructions being at least one of:

(i) a program instruction generating a result value not used by another portion of said executable computer program; and

(ii) a program instruction dependent upon an uninitialised variable;

detecting said executable computer program as containing a computer virus if a number of suspect program instructions identified for said executable computer program exceeds a threshold level;

maintaining a dependence table indicating dependency between state variables within said computer and loaded variable values, wherein for each program instruction a determination is made as to which state variables are read and written by that program instruction and, for each loaded variable value within said dependence table, if any state variable read by that program instruction is marked as dependent upon said loaded variable value, then all state variables written by that program instruction are marked as dependent upon said loaded variable value with previous dependencies being cleared; and

parsing said executable computer program for suspect program instructions by following execution flow and upon occurrence of a branch first following a first branch path having saved pending analysis results and subsequently returning to follow a second branch path having restored said pending analysis results;

wherein a state variable is marked as initialised upon occurrence of one of:

(i) a write to said state variable of a determined initialised value; and

(ii) use of said state variable as a memory address value by a program instruction;

wherein a branch path stops being followed when one of the following occurs:

(i) there are no further suitable program instruction for execution within that branch path; and

(ii) said branch path rejoins a previously parsed execution path;

wherein an initialisation table is included for indicating which state variables have been initialised;

wherein said dependence table includes a column for each register within a processor, an external write, and a write to a flag value; and a plurality of rows with each row corresponding to a value loaded into said computer that influences a state of said computer;

wherein said initialisation table includes a column for each register within said processor indicating whether each register is initialised.

14. (original) A method as claimed in claim 13, wherein said computer virus is a polymorphic computer virus.

15. (cancelled)

16. (currently amended) A method as claimed in claim 13, wherein for each program instruction a determination is made as to which state variables are read by that program instruction to produce a read-mask.

17. (currently amended) A method as claimed in claim 13, wherein for each program instruction a determination is made as to which state variables are written by that program instruction to produce a write-mask.

18. (cancelled)

19. (previously presented) A method as claimed in claim 13, wherein said state variables include at least one of:

(i) register values;

(ii) processing result flag values; and

(iii) a flag indicative of a write to a non-register storage location.

20. (cancelled)

21. (cancelled)

22. (cancelled)

23. (cancelled)

24. (previously presented) A method as claimed in claim 13, wherein if said threshold level is exceeded, then further virus detection mechanisms are triggered to confirm the presence of a computer virus.

25. (currently amended) Apparatus for detecting an executable computer program containing a computer virus, said apparatus comprising:

an analyser for analysing program instructions forming said executable computer program to identify suspect program instructions being at least one of:

(i) a program instruction generating a result value not used by another portion of said executable computer program; and

(ii) a program instruction dependent upon an uninitialised variable; and

a detector operable to detect said executable computer program as containing a computer virus if a number of suspect program instructions identified for said executable computer program exceeds a threshold level, wherein said analyser includes a dependence table indicating dependency between state variables within said computer and loaded variable values, wherein for each program instruction said analyser makes a determination as to which state variables are read and written by that program instruction and for each loaded variable value within said dependence table if any state variable read by that program instruction is marked as dependent upon said loaded variable value, then all state variables written by that program instruction are marked as dependent upon said

loaded variable value with previous dependencies being cleared, and said analyser parses said executable computer program for suspect program instructions by following execution flow and upon occurrence of a branch first following a first branch path having saved pending analysis results and subsequently returning to follow a second branch path having restored said pending analysis results;

wherein a state variable is marked as initialised upon occurrence of one of:

(i) a write to said state variable of a determined initialised value; and

(ii) use of said state variable as a memory address value by a program instruction;

wherein a branch path stops being followed when one of the following occurs:

(i) there are no further suitable program instruction for execution within that branch path; and

(ii) said branch path rejoins a previously parsed execution path;

wherein an initialisation table is included for indicating which state variables have been initialised;

wherein said dependence table includes a column for each register within a processor, an external write, and a write to a flag value; and a plurality of rows with each row corresponding to a value loaded into said computer that influences a state of said computer;

wherein said initialisation table includes a column for each register within said processor indicating whether each register is initialised.

26. (original) Apparatus as claimed in claim 25, wherein said computer virus is a polymorphic computer virus.

27. (cancelled)

28. (currently amended) Apparatus as claimed in claim 25, wherein for each program instruction said analyser makes a determination as to which state variables are read by that program instruction to produce a read-mask.

29. (currently amended) Apparatus as claimed in claim 25, wherein for each program instruction said analyser makes a determination as to which state variables are written by that program instruction to produce a write-mask.

30. (cancelled)

31. (previously presented) Apparatus as claimed in claim 25, wherein said state variables include at least one of:

(i) register values;

(ii) processing result flag values; and

(iii) a flag indicative of a write to a non-register storage location.

32. (cancelled)

33. (cancelled)

34. (cancelled)

35. (cancelled)

36. (previously presented) Apparatus as claimed in claim 25, wherein if said threshold level is exceeded, then further virus detection mechanisms are triggered to confirm the presence of a computer virus.

37. (new) A computer program product as claimed in claim 1, wherein if two different branch paths reach a common point, said different branch paths are treated as separate branch paths.

38. (new) A computer program product as claimed in claim 1, wherein upon reaching a conditional jump, said dependence table and said initialisation table are buffered; and upon reaching a target point of said conditional jump, a current version of said dependence table and said initialisation table and a previously buffered version of said dependence table and said initialisation table are merged.

39. (new) A computer program product as claimed in claim 1, wherein upon reaching a loop, said dependence table and said initialisation table are buffered, where said dependence table and said initialisation table are merged during a next pass through said loop.